MFT Standalone Tracking:
A Track Finding Algorithm using a Cellular Automaton

ITS/MFT Week, March 10-12, 2014, CERN

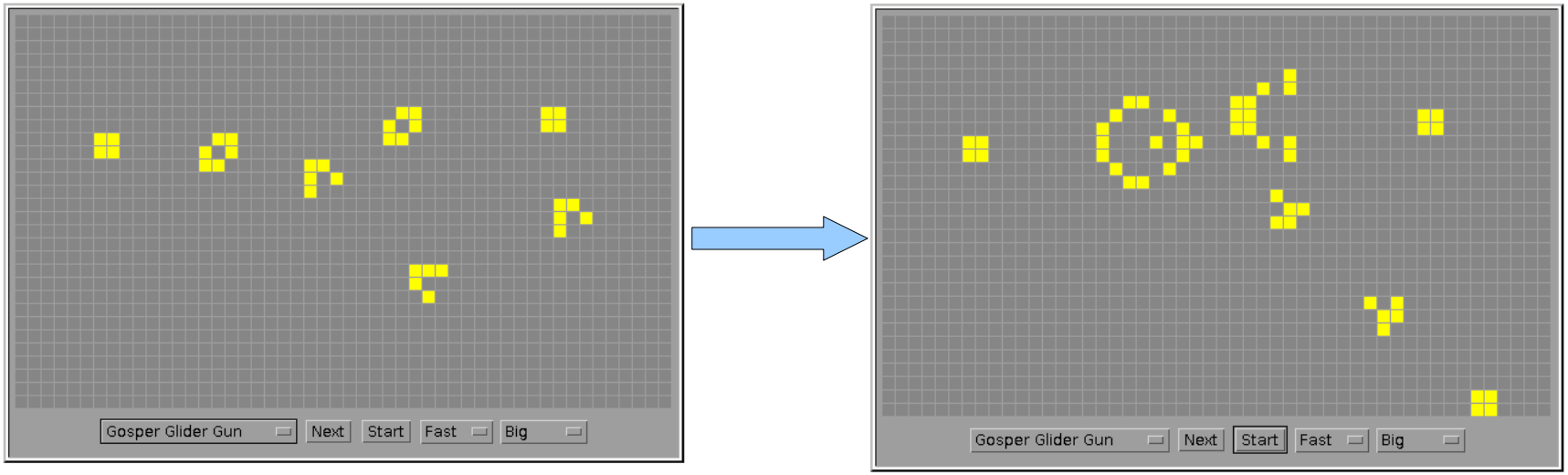
Bogdan Vulpescu
Laboratoire de Physique Corpusculaire
Clermont-Ferrand, France

The plan

- a little history of the cellular automaton and its later applications to particle detectors
- implementation of a cellular automaton track finding algorithm for a detector toy model and simple Monte-Carlo simulations (outside AliRoot)
- performance of the track finding: preliminary results, versions of the MFT geometry, etc.
- outlook: towards standalone track finding (+ tracking, Kalman filter) in AliRoot

History

- John von Neumann: self-replicating automata (field of cellular automata), 1940, '50, '60
- John Horton Conway: 1970 Game of Life = a zero player game



www.bitstorm.org/gameoflife

- Stephen Wolfram ('80): cellular automaton ---> computationally universal Turing machine
- T.Toffoli, N.Margolus, Cellular Automata Machines - A new environment for modeling
  The MIT Press 1987
  "Cellular automata are *discrete* dynamical systems whose behavior is *completely* specified in terms of a *local* relation [...]."

Applications for particle physics detectors

A.Glazov, et al.
Filtering tracks in discrete detectors using a cellular automaton, N.I.M. A329 (1993) 262
ARES spectrometer, MWPC

M.P.Bussa et al.
On a possible second-level trigger for the experiment DISTO, Il Nuovo Cimento Vol. 109 A, N. 3 (1996)
spectrometer, scintillators (fibers) + MWPC

I.Kisel et al.
Cellular automaton and elastic net for event reconstruction in the NEMO-2 experiment, N.I.M. A387 (1997) 433
scintillators, Geiger cells

I.Abt et al.
HERA-B, N.I.M. A489 (2002) 389 and A490 (2002) 546
CATS: Cellular Automaton Tracking in Silicon (vertex detector)

GSI group (University of Frankfurt, UniCA code) ---> STAR HFT, CBM, ITS

The ALICE/MFT geometry with 5 planes

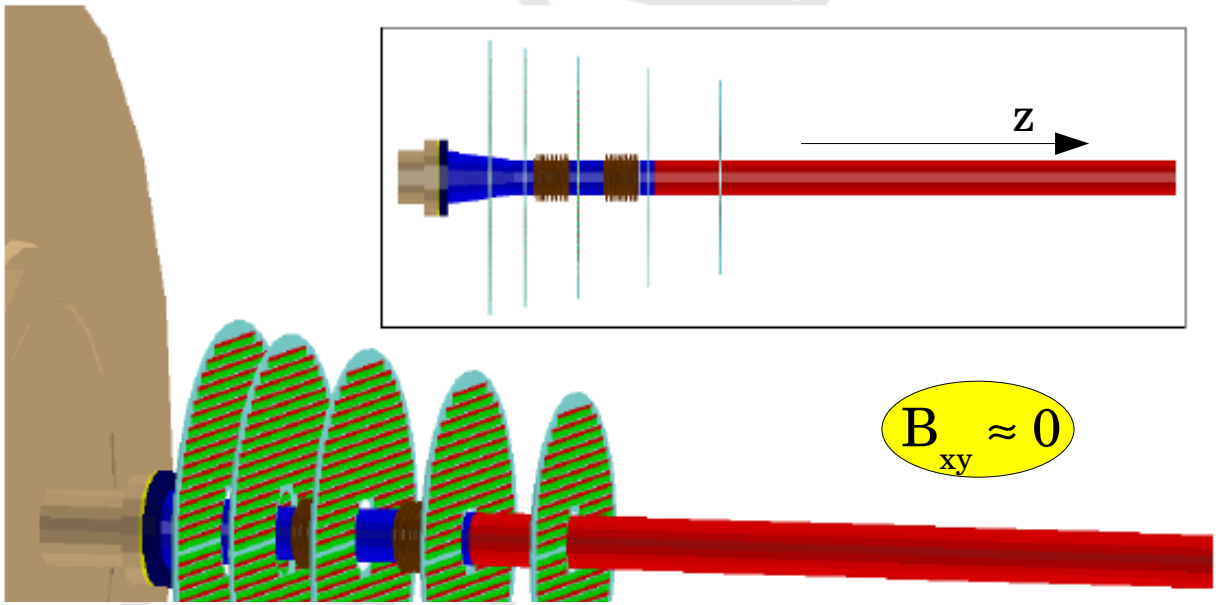


Table 2.2: *Dimensions and positions for the MFT planes in the MFT Standard Setup.*

Plane	Int. radius (cm)	Ext. radius (cm)	Z location (cm)	Pixel pitch (μm)	Thickness (% of X_0)
0	2.5	11.0	-50		
1	2.5	12.3	-58		
2	3.0	13.7	-66	25	0.4
3	3.5	14.6	-72		
4	3.5	15.5	-76		

From the MFT LOI 2013 (5 planes)

A toy model for the MFT detector (ROOT)

detector = plane surface

track = straight line + multiple scattering in detector (gaussian small angle)

hit = intersection point between a line and a plane (a single pixel, σ = 7 μm)

noisy pixels = uniformly distributed in the plane (prob. = 1.E-5)

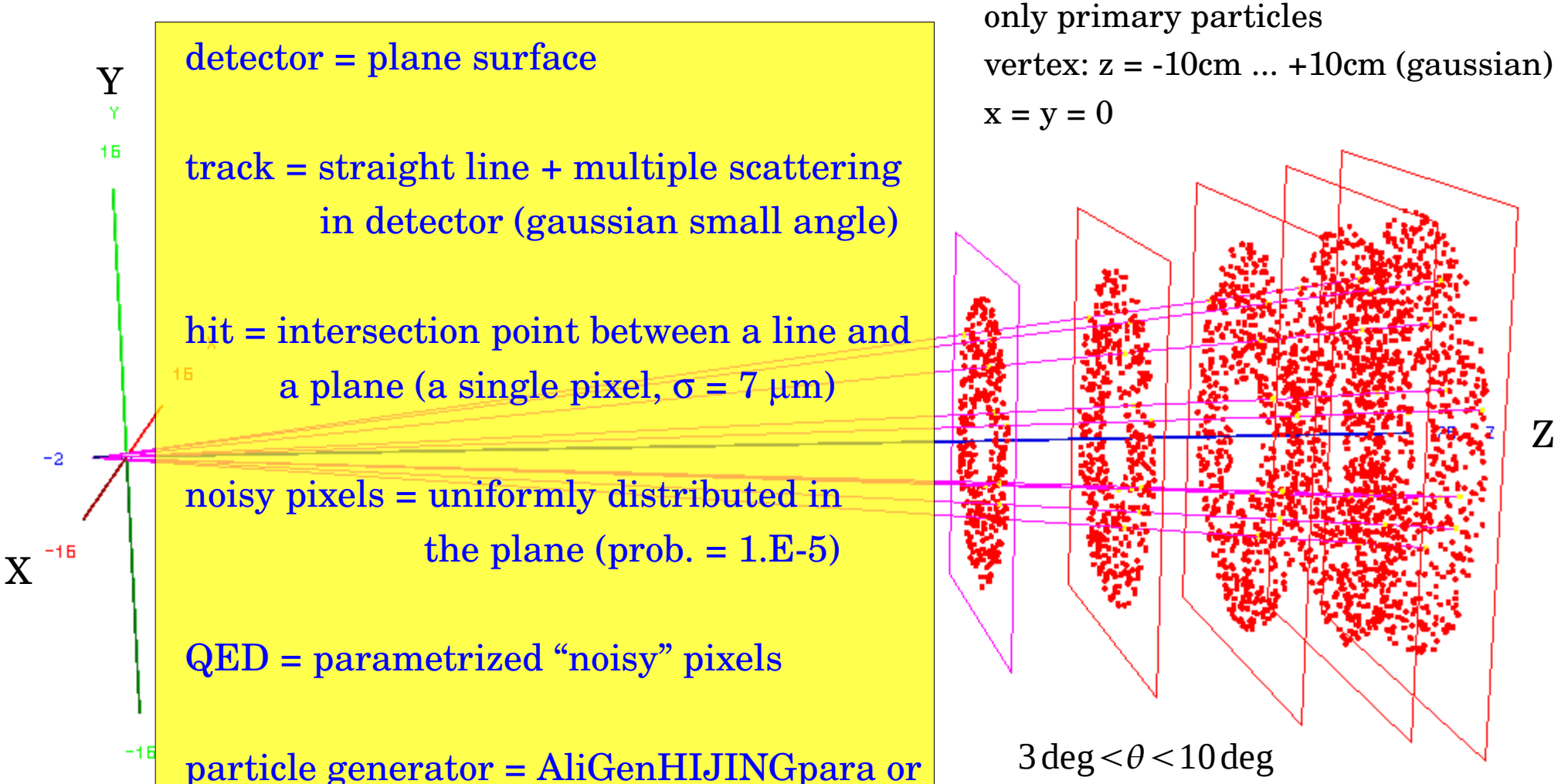
QED = parametrized “noisy” pixels

particle generator = AliGenHIJINGpara or input from AliRoot clusters

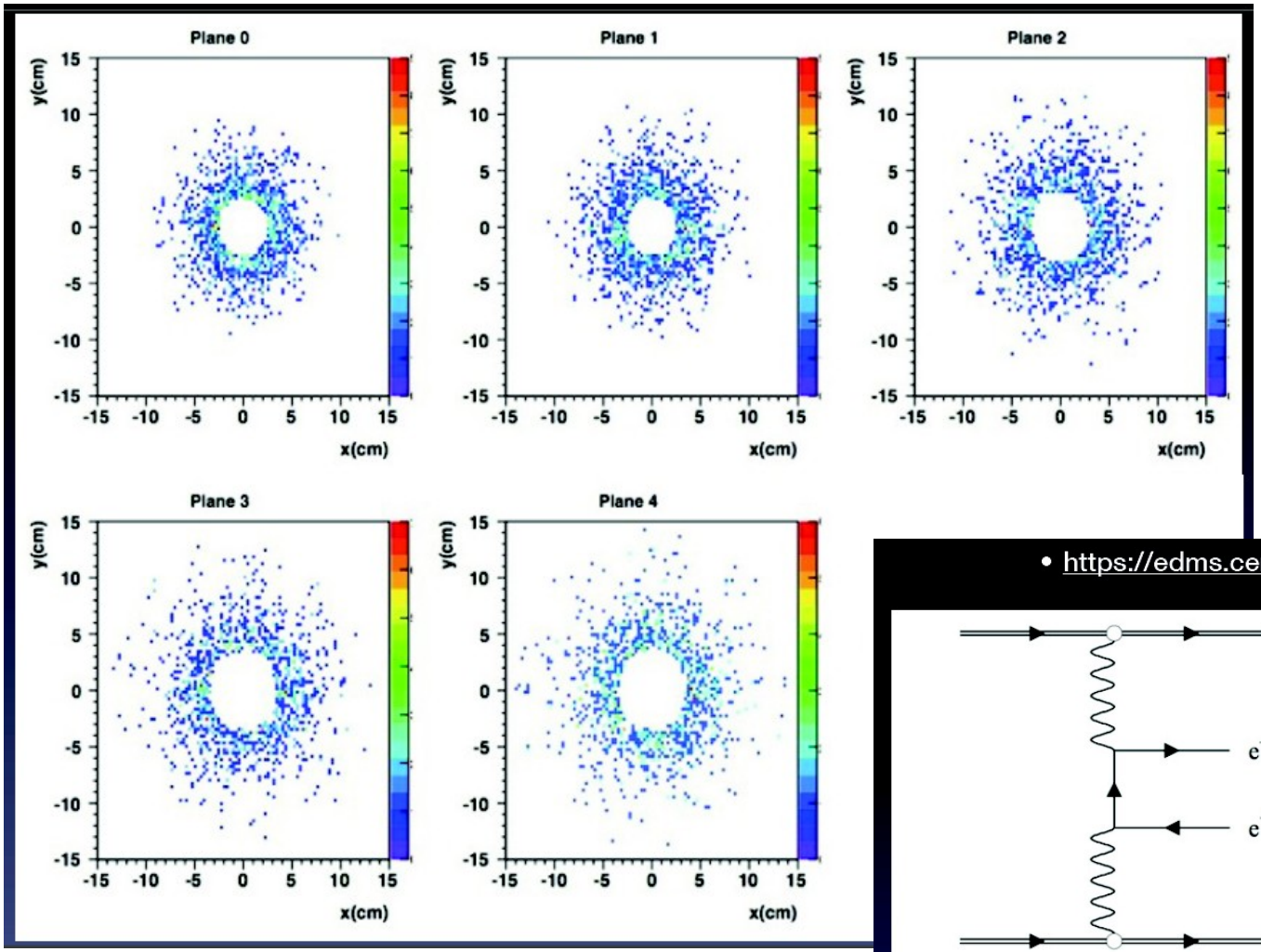
only primary particles

vertex: z = -10cm ... +10cm (gaussian)

x = y = 0



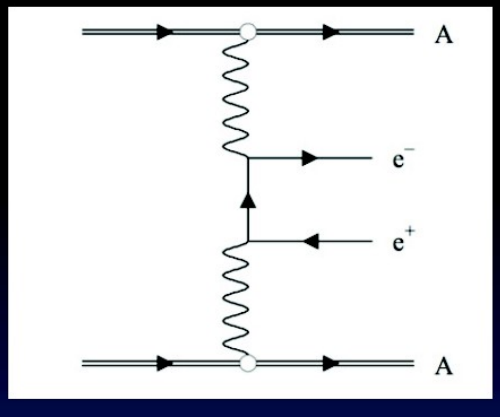
Hits from QED processes (Raphael Tieulent, IPNL)



AliRoot + QED generator + MFT ---> parametrization of the hit densities

- uncorrelated hits (p < 100 keV)
- depends on the luminosity and on the CMOS integration time
- Ncls(QED) ~ 0.5 x Ncls(had)

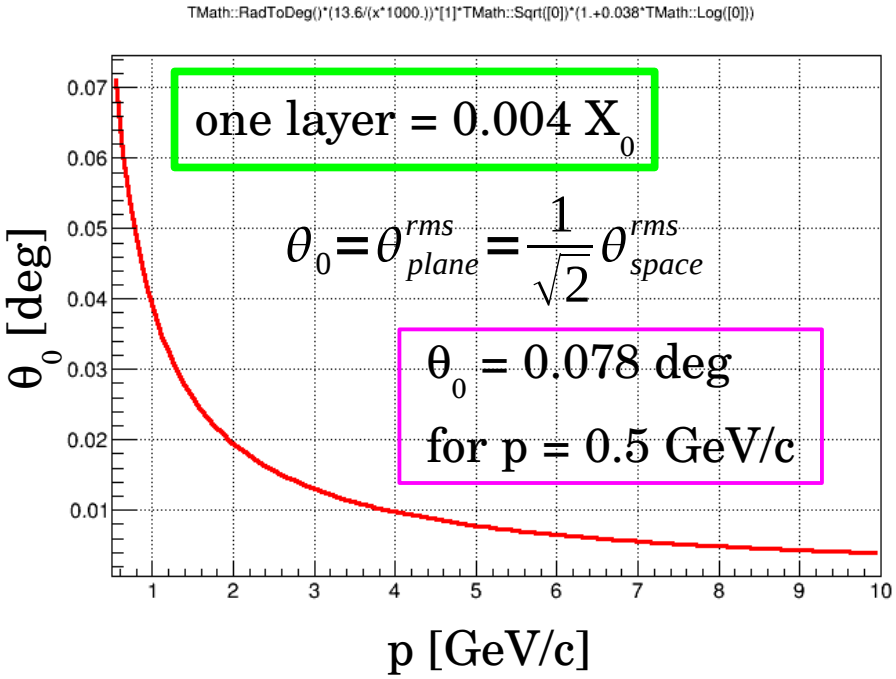
- https://edms.cern.ch/file/357571/1/ALICE-INT-2002-027.pdf



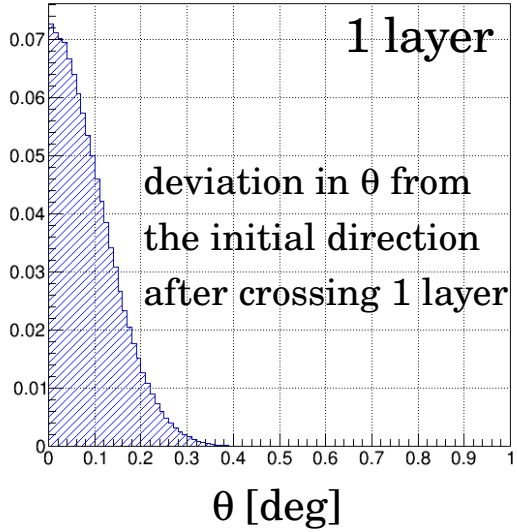
$N(e^+e^-)$	Cross section
1	223 kb
2	4.0 kb
3	0.78 kb
4	0.22 kb
5	0.07 kb

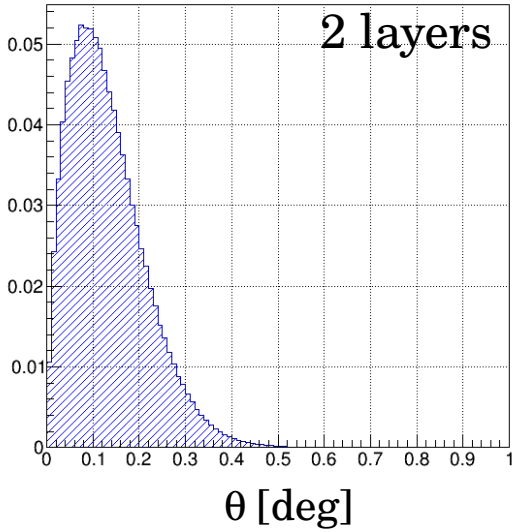
radial hit density histograms

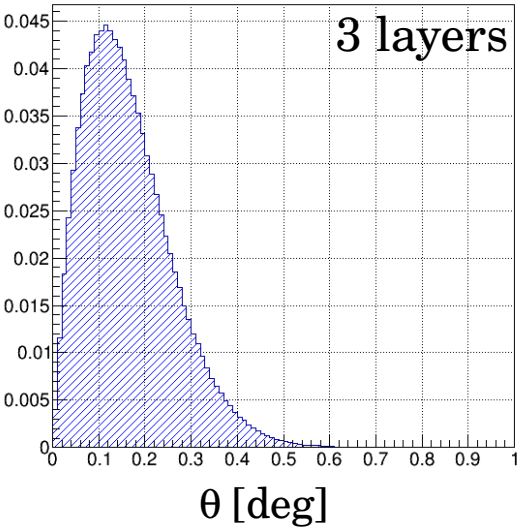
Multiple scattering in the detection planes

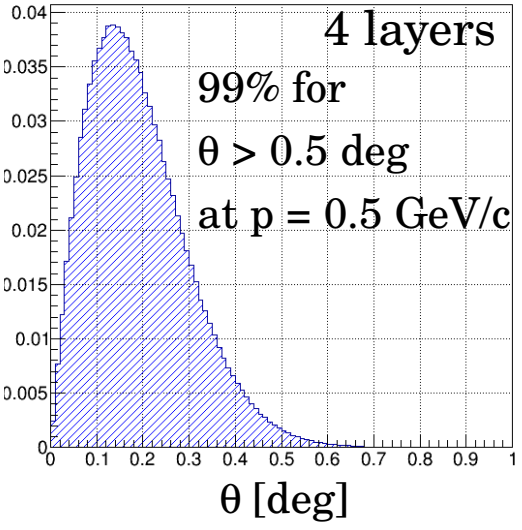


$$\theta_0 = \theta_{plane}^{rms} = \frac{1}{\sqrt{2}}\theta_{space}^{rms}$$









Simulation:

at each passage through a plane

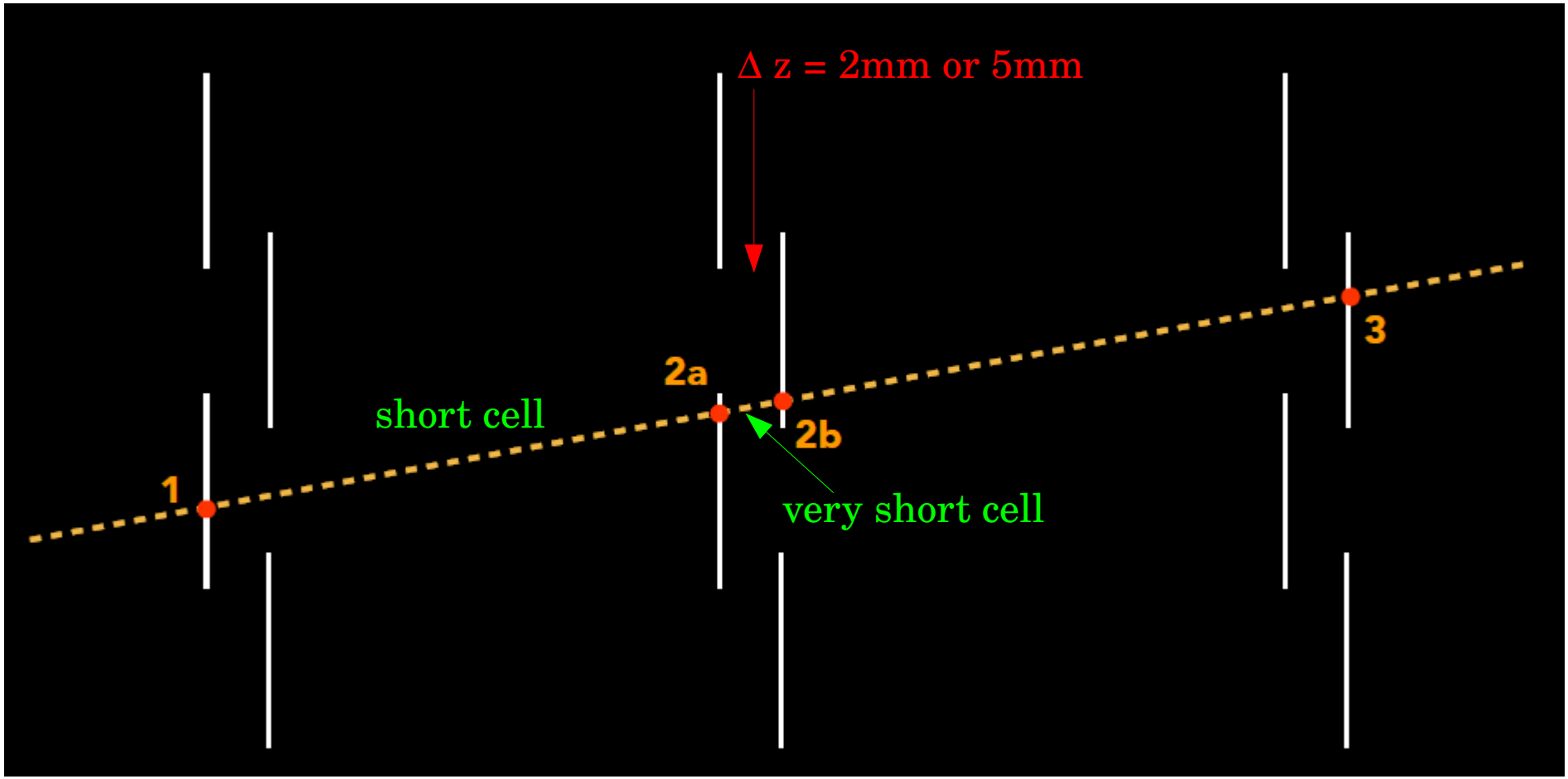
the change in direction is calculated as:

- φ = uniform [0,2π)
- θ = gaussian

Track finding: fixed or adaptive cuts

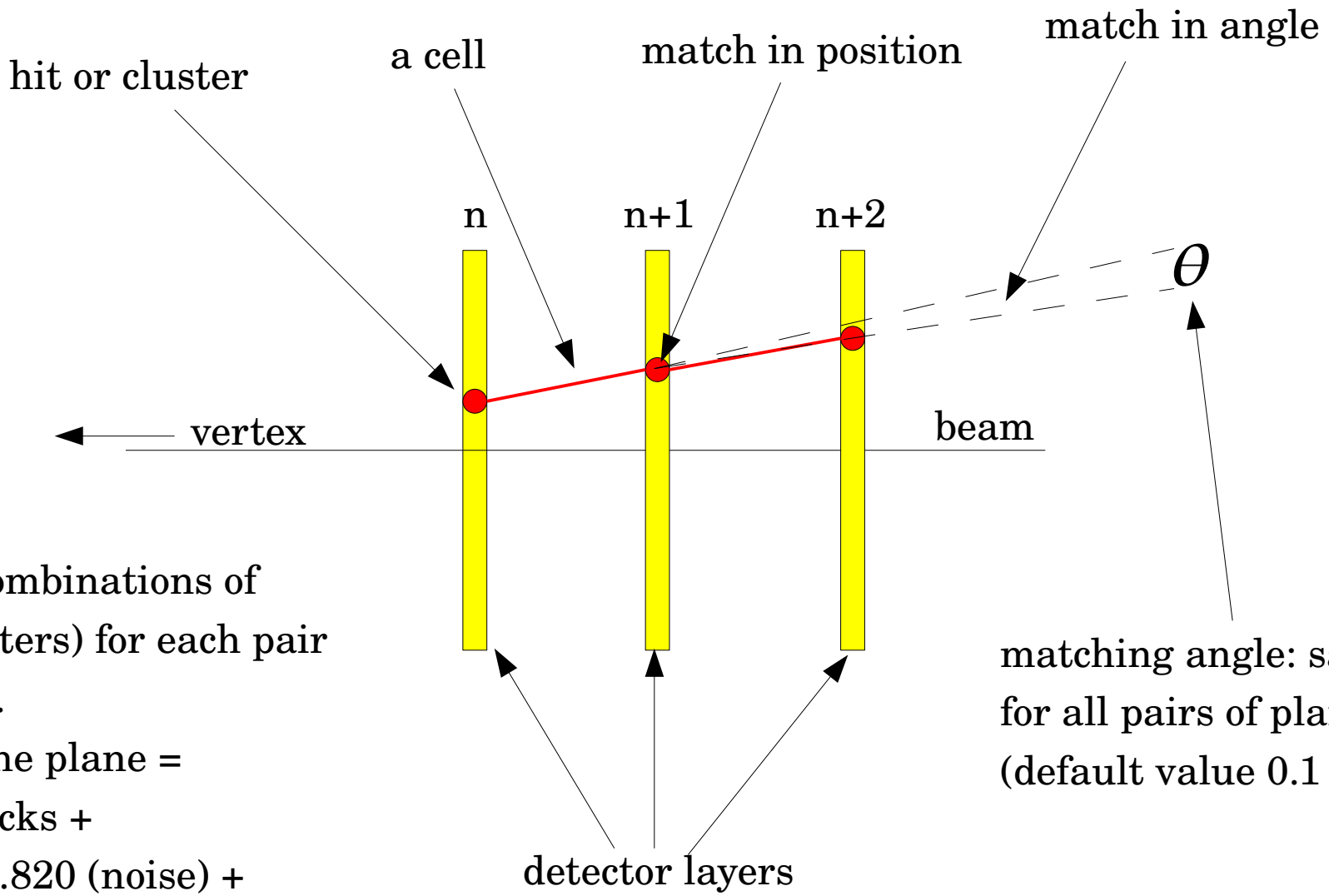
Using as input clusters from AliRoot simulation
(Raphael Tieulent, IPNL)

more complex geometry of the planes: “very short” (ladders overlap), “short” and “long” cells



(in study)

Building a cell; matching the cells



Try all combinations of hits (clusters) for each pair of planes.

Hits in one plane =

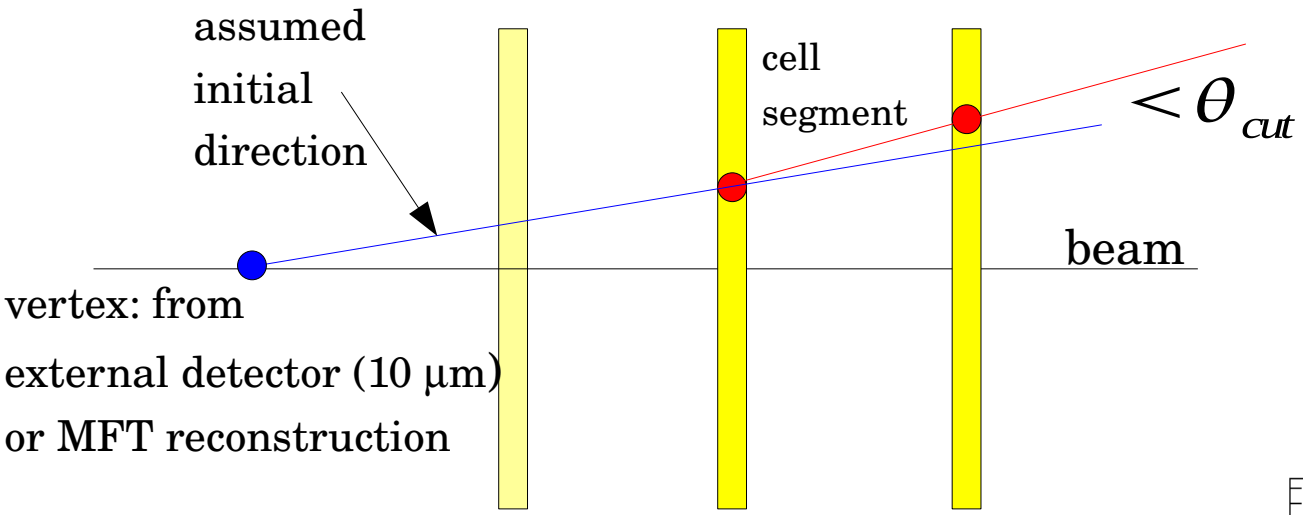
Ntracks +

350...820 (noise) +

750...900 (QED)

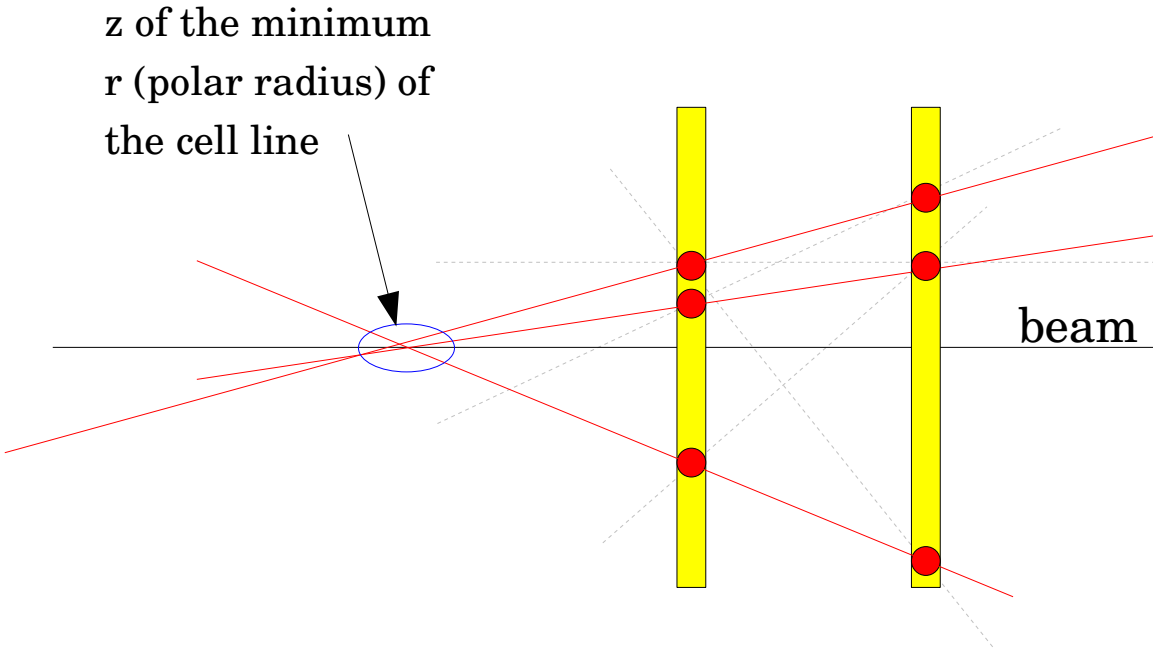
matching angle: same for all pairs of planes (default value 0.1 deg)

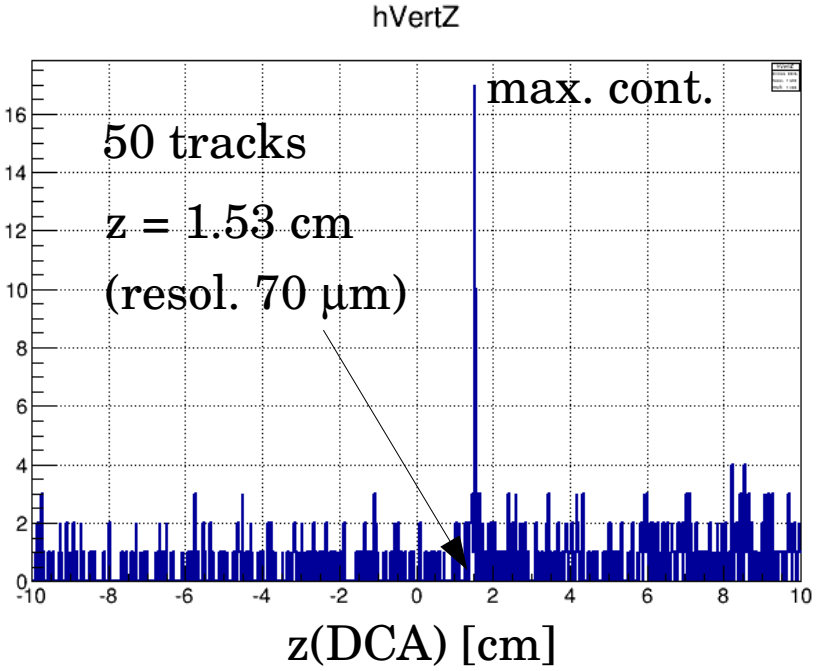
Keep all cells ? No, use the vertex as selection criterion



depending on the plane position (adaptive) or fixed

- the cells between the first two planes (θ < 15 deg) are used to find the z position of the vertex



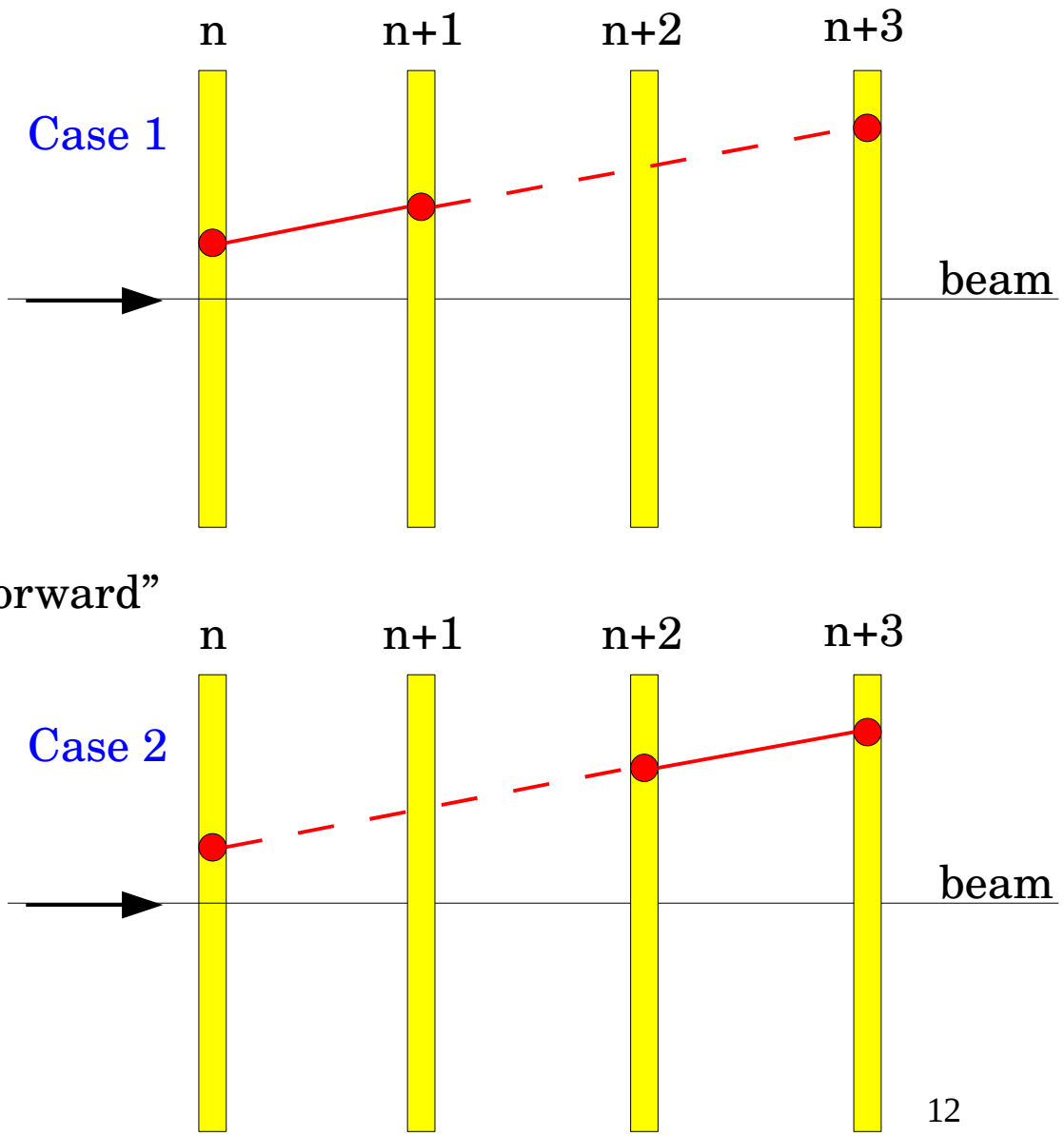


Treating the missing efficiency of one plane

- after the first “RunForward” (see later description) using only the “short” cells, try to connect the cells remained isolated up- (Case 2) or down-stream (Case 1) to “long” cells, counting for one missing hit

- then reset the cell status and redo “RunForward” followed by “RunBackward”, etc.

- two or more subsequent inefficient layers are not considered in this algorithm



General rules of cellular automata

A cellular automaton is a discrete dynamic system whose behavior is completely determined by local mutual relations of states of these systems [...].

discrete = cells with states (usually 0 and 1)

dynamic = evolution in discrete time steps following a set of rules

local mutual relations = the state of one cell changes according to the states of its neighbors (first order neighbor)

clocked system = the cell states change simultaneously

In tracking detectors, two approaches:

one cell is one hit (cluster)

one cell is one segment connecting two close detection planes

Rules follow physics and are adaptive:

vertex production

magnetic field

multiple scattering

detector efficiency

detector noise

Operations flow: CreateCells, RunForward, RunBackward

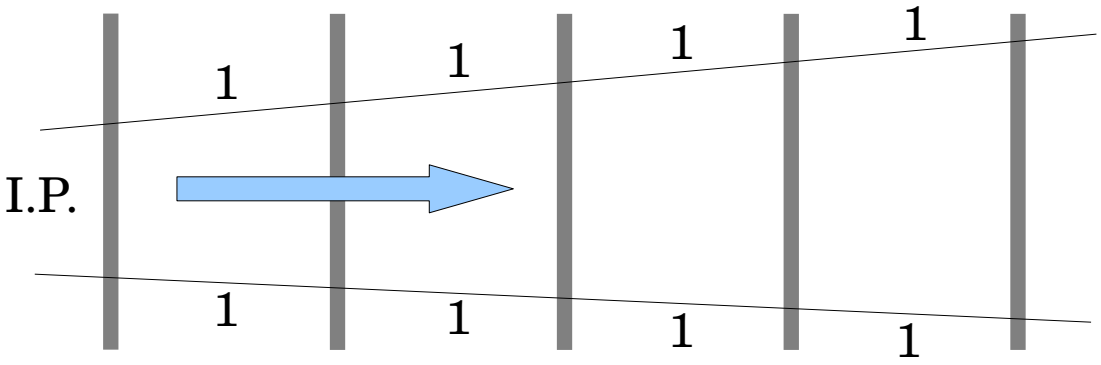
- create cells
  - from hits to segments of hits, with some selection rules to decrease the number of combinations to be analyzed in the next step

- run forward (from the vertex towards the end of the detector)
  - apply rules for matching neighboring cells, establish neighboring relations between cells

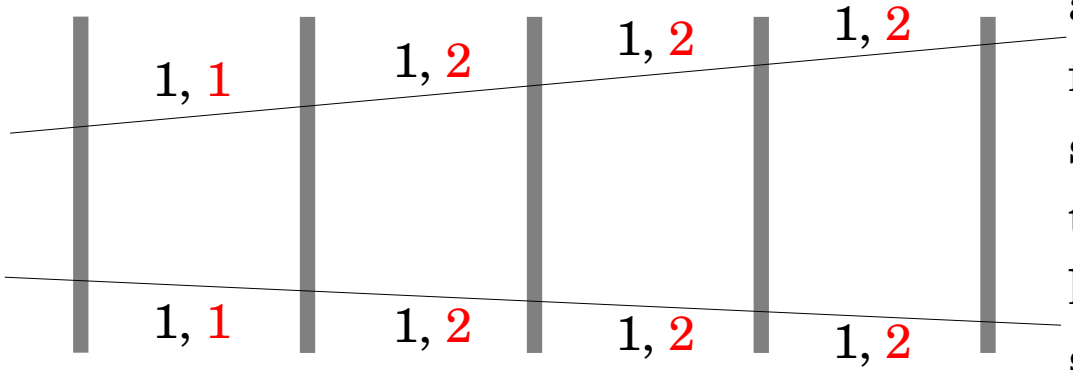
- run backward (from the end of the detector towards the vertex)
  - search for the longest *contiguous and ordered* succession of cells

- custom filtering: eliminate split tracks, chi-square selection, etc.

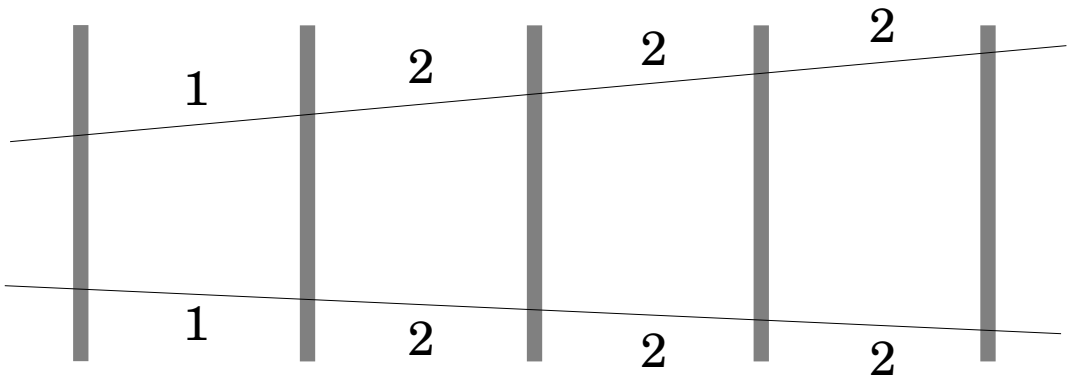
The forward evolution with two tracks: 1st iteration



status of the cells before the 1st iteration; after creating the cells, status equals by default to 1

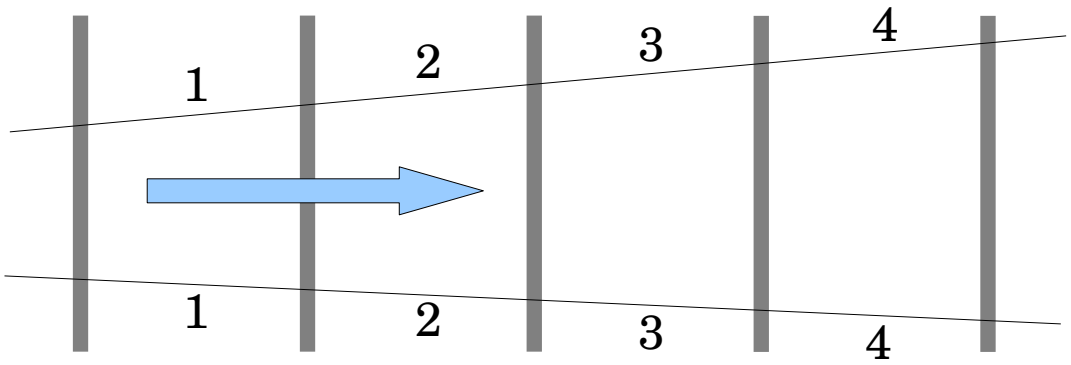


at the end of the first iteration of the forward evolution, the *to-be-updated* status is marked in red;
the status will increase for those cells which have a matching left neighbor with equal status

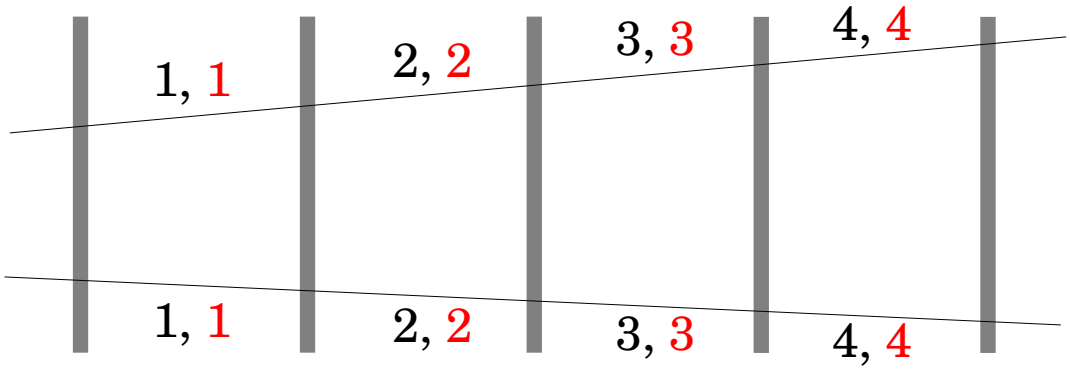


synchronously, all cells now effectively update their status according to the values found at the end of the first iteration

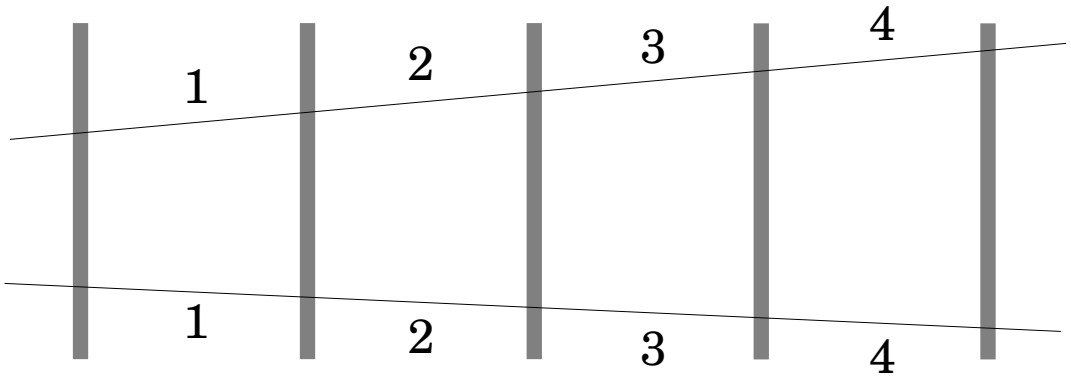
The forward evolution with two tracks: 4th iteration



status of the cells before the 4th iteration

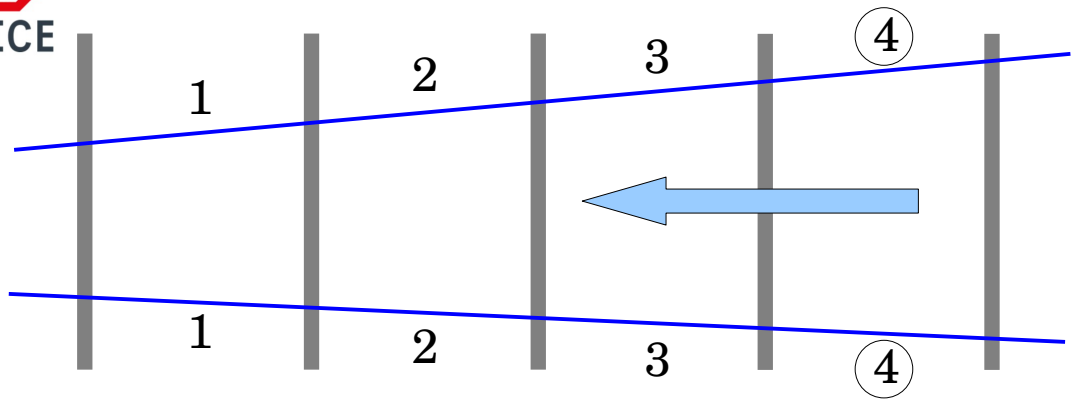


this iteration just checks that the values of the status are frozen and consequently no further iteration is necessary



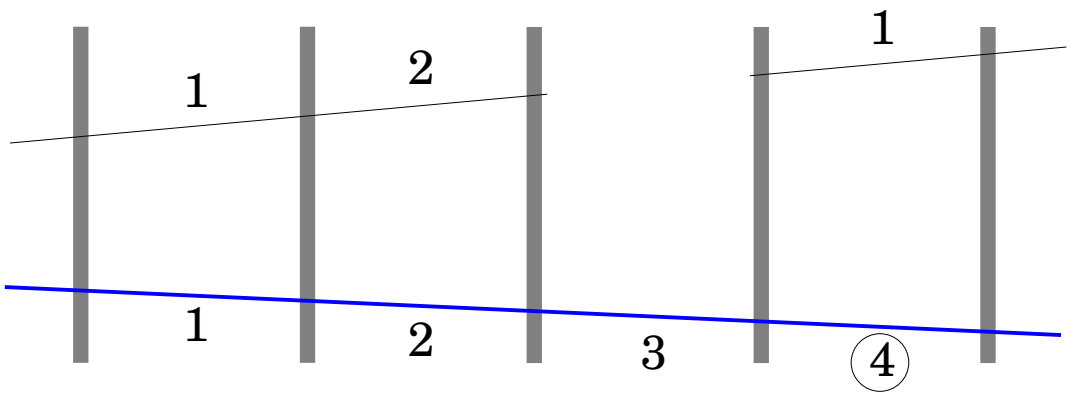
status of the cells at the end of the forward evolution

Find tracks

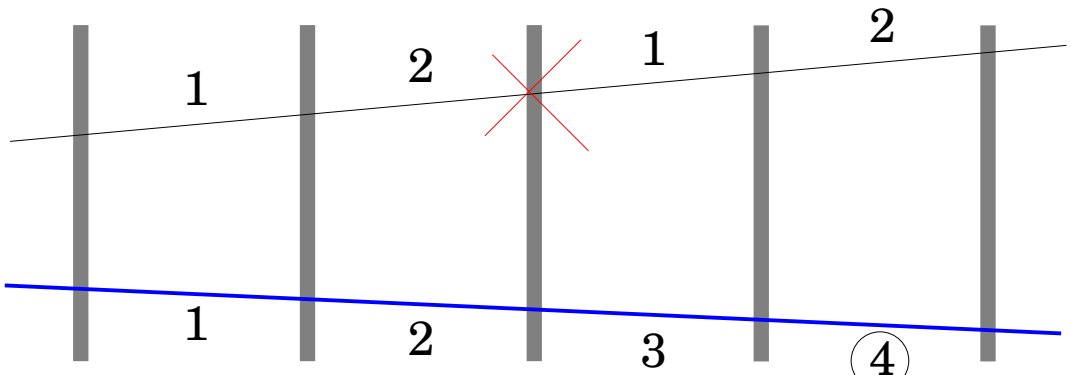


in the *backward evolution*, both tracks are found with 4 cells

Other cases



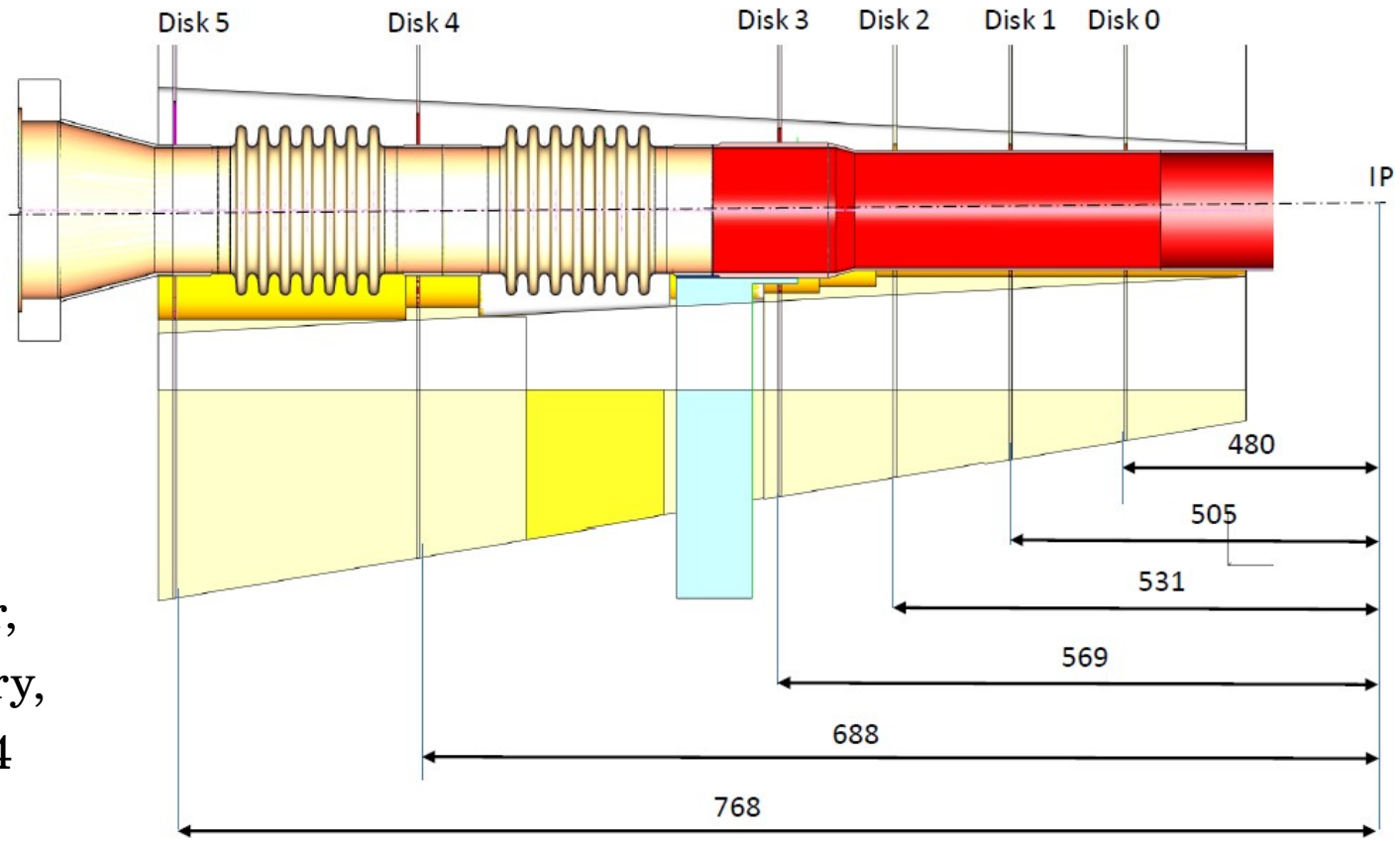
one cell is missing due to vertex selection



two cells did not match in direction

MFT geometries in study: 5 planes, 6 planes

Proposed distances between each disc and the IP

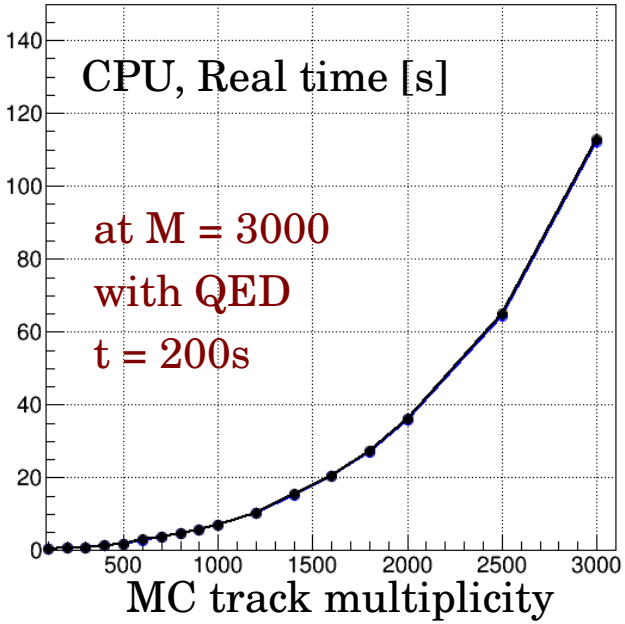


6 planes:
G Batigne,
J-M Buhour,
MFT Plenary,
18 Feb. 2014

Beam pipe:
ϕ = 39.6mm
(or 36mm)
d = 0.8mm
Beryllium

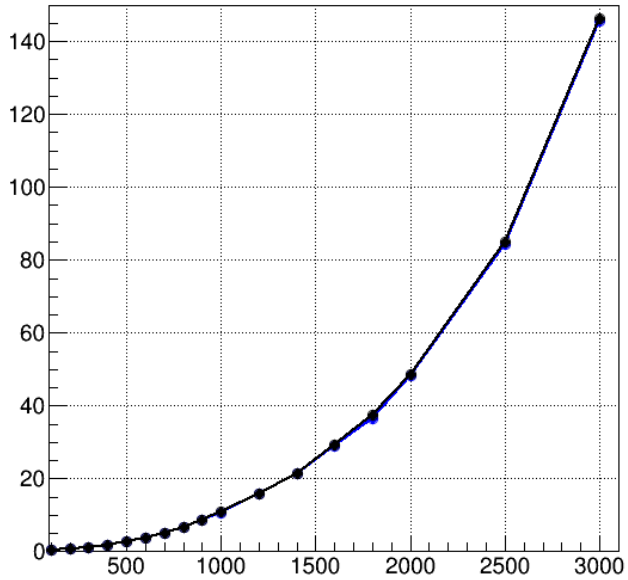
The computing time in the CA algorithm

Only the time spent in the CA routines is counted

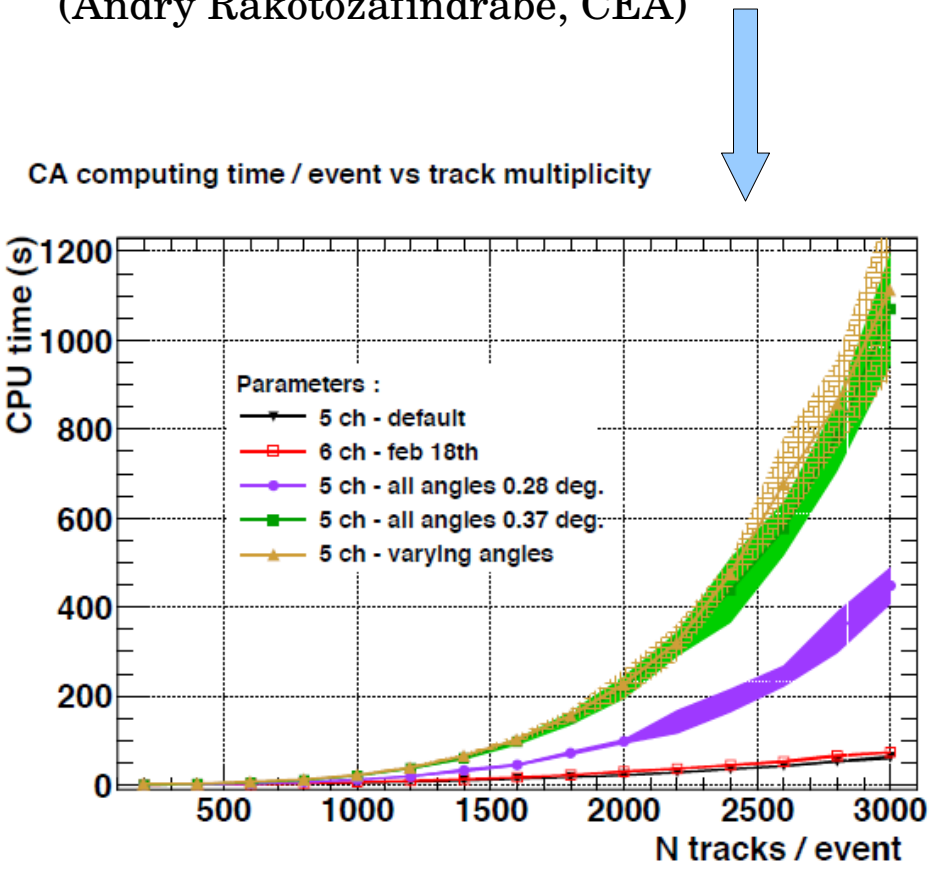


MFT geometry with 5 planes (no QED)

MFT geometry with 6 planes (no QED)

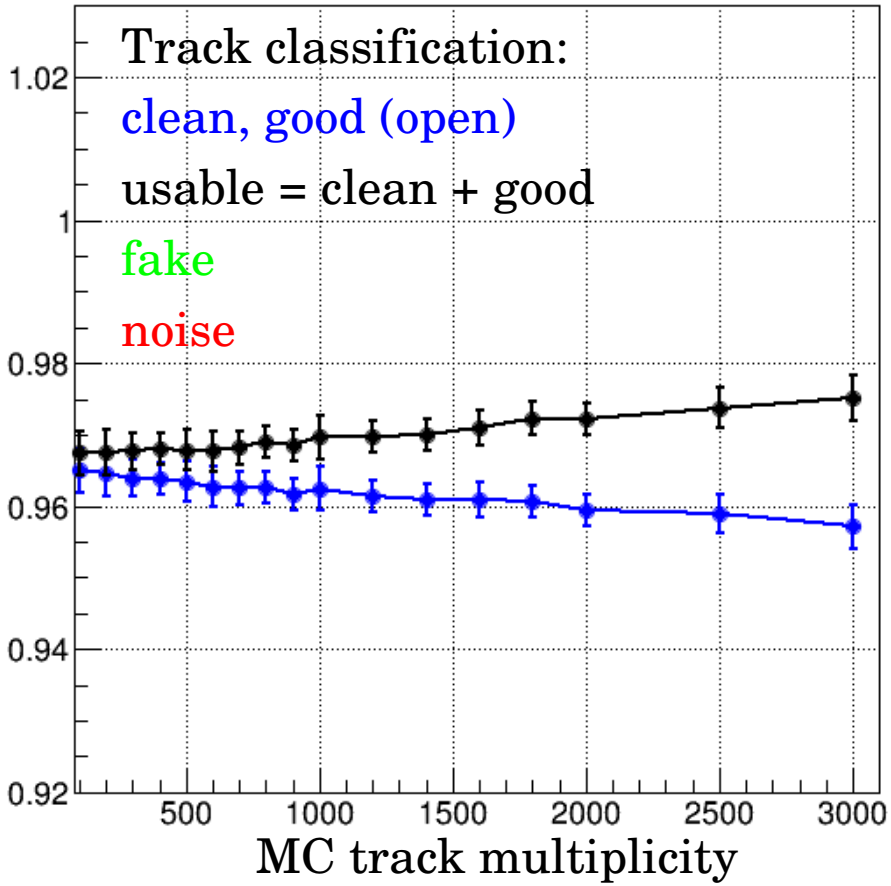


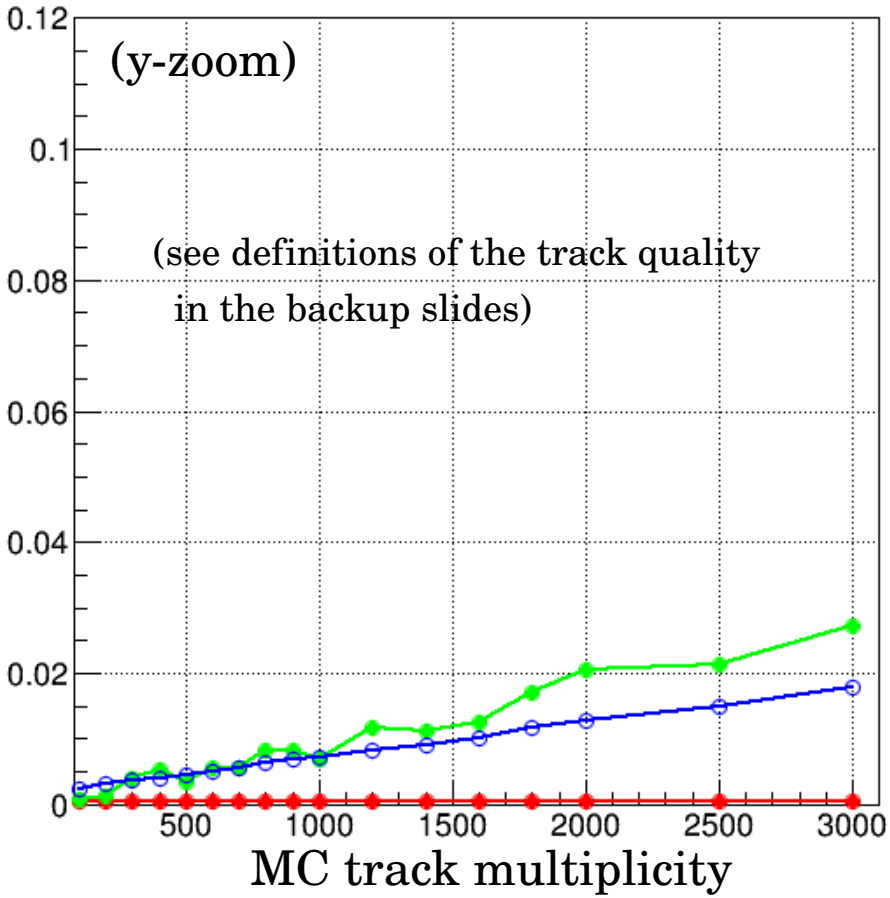
by varying the "searching path" angular parameter, the number of selected cells increases rapidly (Andry Rakotozafindrabe, CEA)


CA computing time / event vs track multiplicity
CPU time (s)
N tracks / event
Parameters :
5 ch - default
6 ch - feb 18th
5 ch - all angles 0.28 deg.
5 ch - all angles 0.37 deg.
5 ch - varying angles

Track finding efficiency

MFT with 5 planes





(split tracks are not yet filtered out)

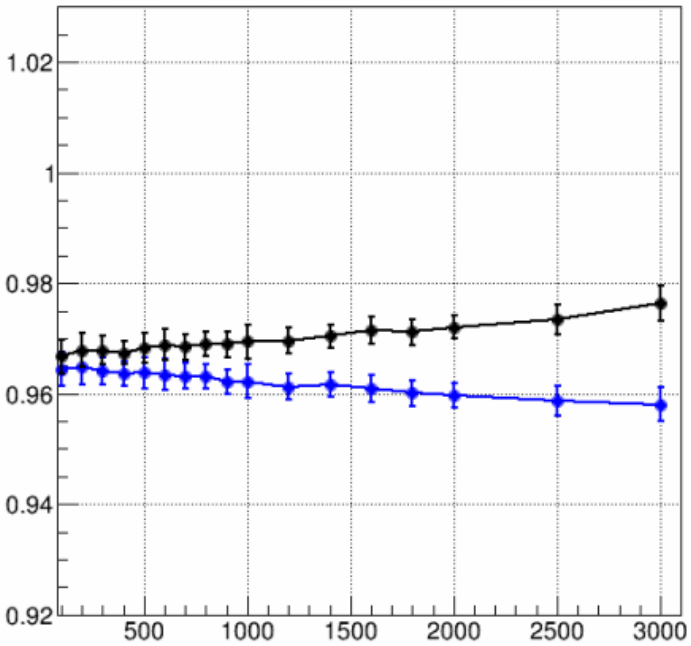
Geometry, noise, QED, ...

same track finding parameters in all cases

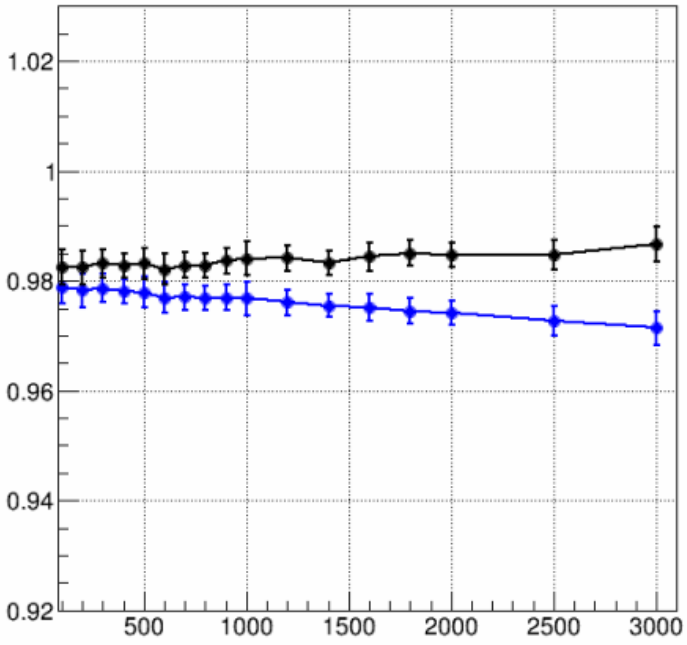
the requested track length is the same (4 planes out of 5 or 6)

the fraction of "good" tracks is increasing by ~1%

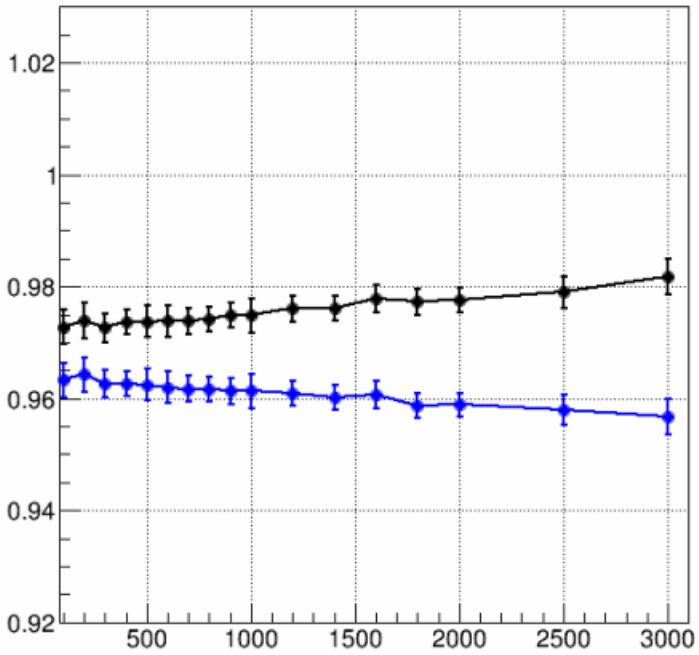
MFT 5 planes, noise 1.E-5, no QED



MFT 6 planes, noise 1.E-5, no QED

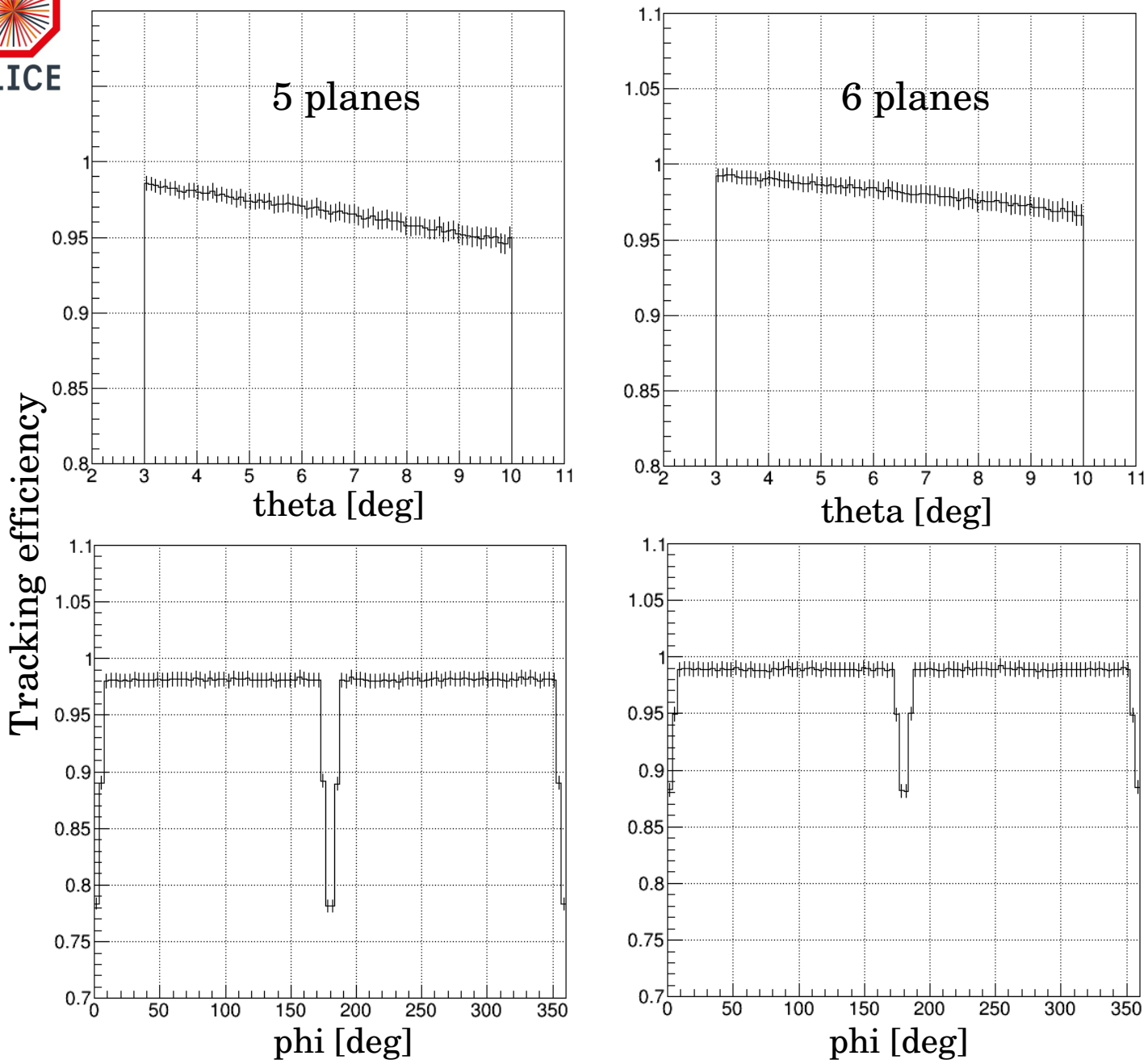


MFT 5 planes, noise 1.E-5, QED



MC track multiplicity

(no beam pipe)



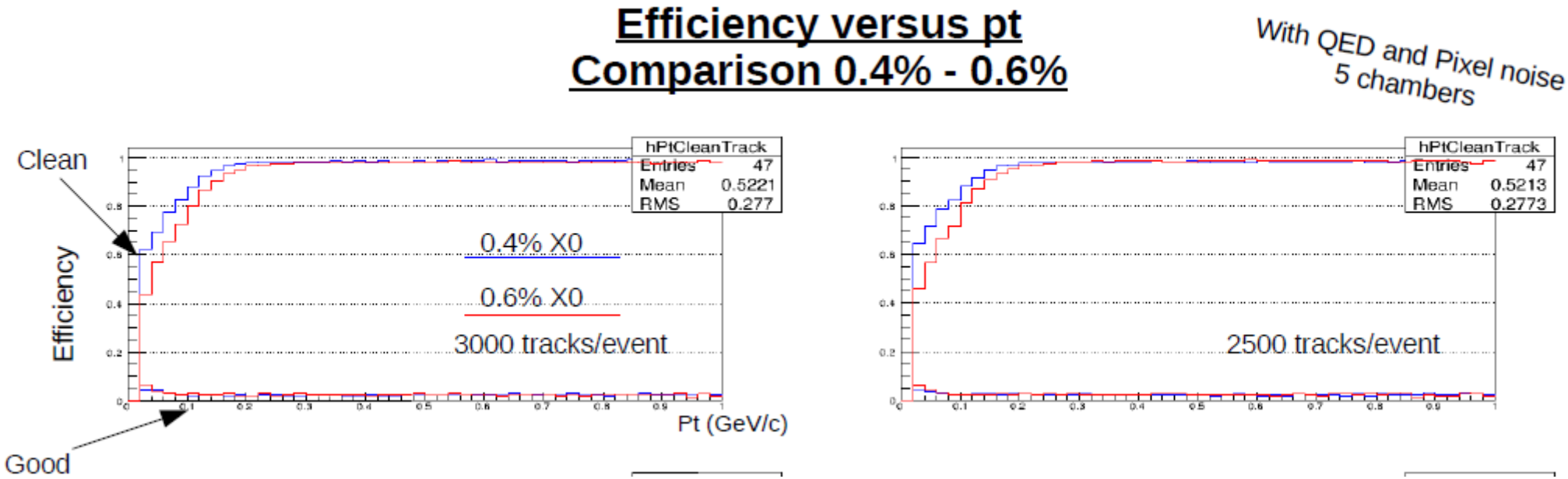
Efficiency as function of the acceptance (all multiplicities, usable tracks = “clean” + “good”)

- at larger theta, low momentum tracks predominate (large diffusion angle), while the opening of the searching path is the same everywhere (0.1 deg)

- at phi = 0 and 180 deg the track finding is sensitive to the beam pipe support (less for 6 planes)

The material budget of one MFT plane
(Franck Manso, LPC)

The multiple scattering, through the total momentum, drives the dependence of the track finding efficiency on the material budget; the pt distribution of the parametrized (HIJING) generator peaks at ~ 0.2 GeV/c



Outlook

- Cellular Automaton track finding in a toy Monte-Carlo MFT simulation works well
- it works with “toy” tracks as well as with clusters generated by AliRoot simulation
- it allows to study different possible geometries of the MFT
- vertex reconstruction: it works well for M >= 200 (pixel noise 1.E-5, QED included) with a simple vertex reconstruction procedure; for M <= 100 needs more study or external vertex
- work in progress to refine the track filtering after the track finding: remove of split tracks, further reduction of fake tracks
- work in progress to treat double hits in the overlap regions of the sensitive ladders
- work in progress to study vertex performance: secondary particles, pile-up events
- next: implementation in AliRoot as a standalone tracking (probably with a Kalman filter on top)

Backup slides

The collision vertex determined from the first layer of cells

- search the minimum of $(x + y^2)$ of the track; the beam is $x = y = 0$

- the equation of the track (straight line) $\frac{x - x_0}{a} = \frac{y - y_0}{b} = \frac{z - z_0}{c}$

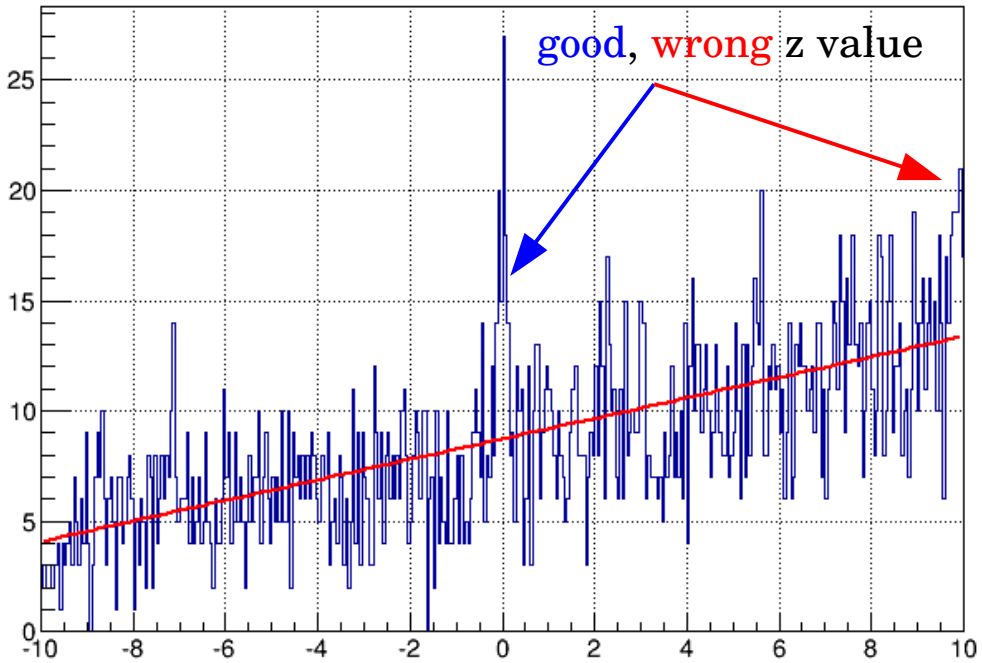
- with the notations $n_1 = \frac{c\,x_0 - a\,z_0}{c}$, $n_2 = \frac{a}{c}$, $n_3 = \frac{c\,y_0 - b\,z_0}{c}$, $n_4 = \frac{b}{c}$

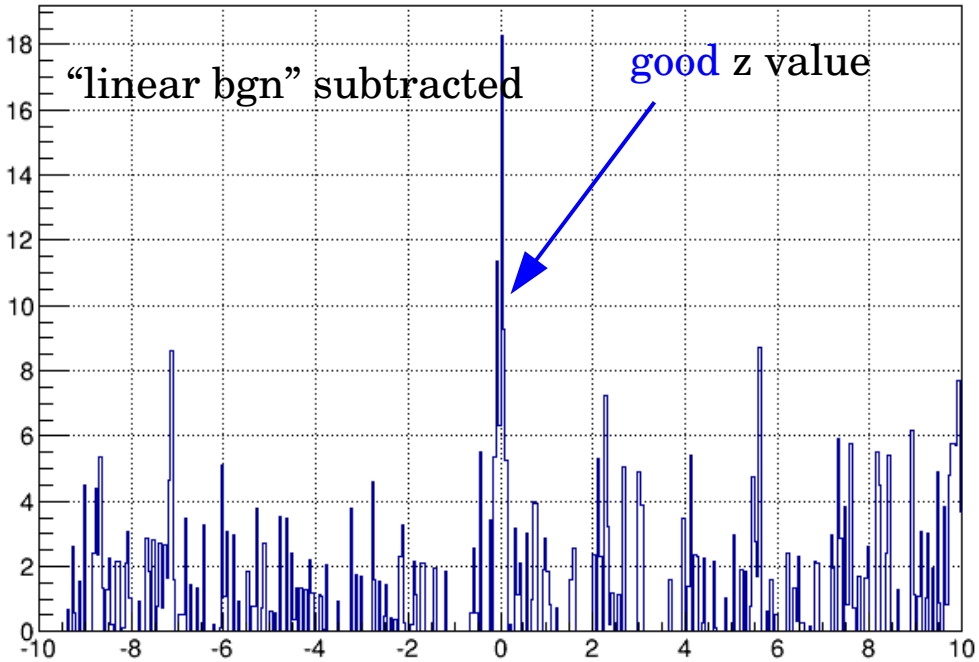
- we have $z|min\,(x^2 + y^2) = -\frac{n_1 n_2 + n_3 n_4}{n_2^2 + n_4^2}$

This z value is accumulated in a histogram and the highest bin content gives the estimated vertex. A similar method is applied in the UniCA code.

Finding the vertex for mult = 100 and QED

- important background from pixel noise and QED combinatorics
- the bin with maximum content is not always enough
- moving integrating window (+/- 2 bins); still possibility to miss the peak (z = 10cm)
- subtract a “linear background” is improving the performance of this method

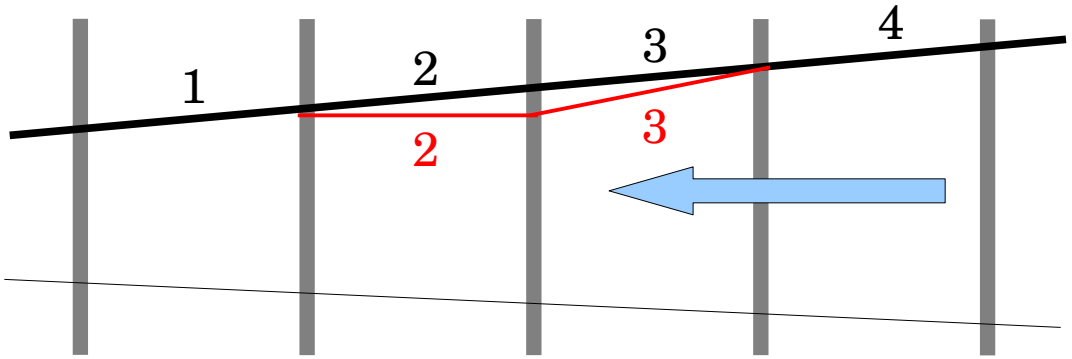




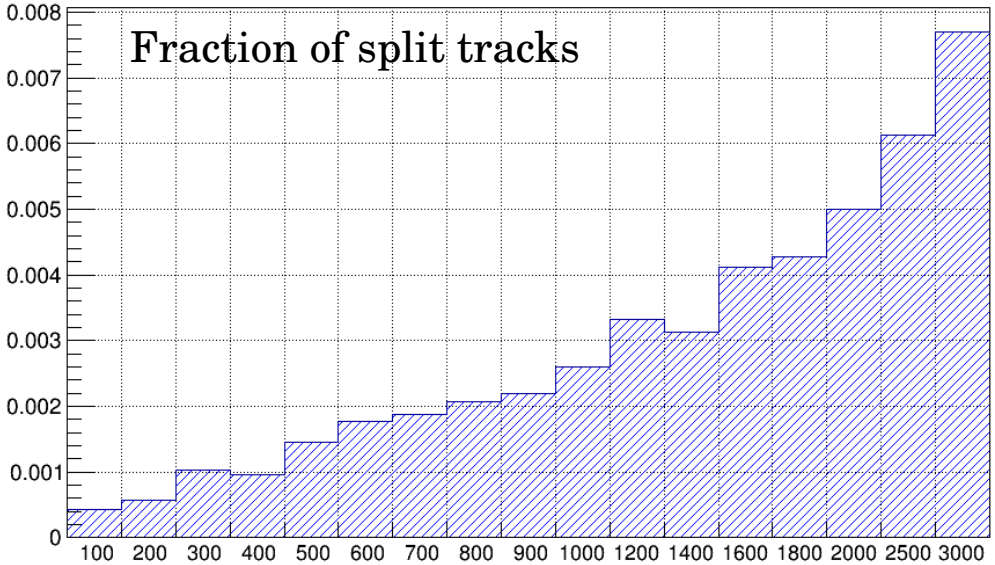
Reconstructed track classification

- the generated (MC) tracks, Nt, have indexes from 0 to (Nt-1)
- a reco track with Nc cells (segments) is made of Nh = (2*Nc) hits
- loop over the reco track hits and count which MC tracks contribute and with how many hits (a noise or QDE hit has by defintion a track index of “-1”)
- if: there is a single track (of index >=0) which contributes to the reco track
  ---> clean track
- else if: the track which contributes with most of the hits has (index >= 0 and) the number of hits >= (Nh-1)
  ---> good track
- else if: the are hits from noise or QED (track index “-1”)
  ---> noise track
- else: there are only hits from “real” tracks, but none of them contributes with >= (Nh-1) hits
  ---> fake track

Split tracks during RunBackward



One type of split tracks:
the two tracks share at least
the first (rightmost) cell



The code on the CERN SVN Server

Web

https://svnweb.cern.ch/cern/wsvn/alicemft/Tracking/CellularAutomaton

Download

svn co svn+ssh://<cern_user_name>@svn.cern.ch/reps/alicemft/Tracking/CellularAutomaton <your_copy>
(see the README file)

Dependencies

ROOT + AliRoot

Current revision

438